Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 135 (2004) 223–233

DISCRETE
APPLIED
MATHEMATICS

www.elsevier.com/locate/dam

Complexity of implementing functions of k-valued logic by circuits and formulas in functionally complete bases ☆

V.A. Orlov[1]

Russian State Humanitarian University, 111116 Moscow, Russia

Abstract

Algorithmic problems are considered that are related to implementing bounded-deterministic functions by circuits and formulas of the minimum size in automaton bases. The problem of finding the asymptotics of the Shannon function is known to be algorithmically undecidable in the case of complete bases, but the coefficient in the formula for the Shannon function can be found with arbitrary accuracy. In the paper the so called strong algorithmic undecidability of the problem of finding the asymptotics of the Shannon function in the case of functionally complete bases is proved. A basis is called cf-equivalent if the constants in the asymptotic formulas for the Shannon function in the classes of circuits and formulas coincide. The existence of bases that are not cf-equivalent is proved in the case of functionally complete bases. It is proved that the recognition problem for the cf-equivalence of a basis is algorithmically undecidable in the strong sense.

Keywords: Algorithmic undecidability; Implementation; Shannon function; Functionally complete bases; k-Valued logic

1. Introduction

The implementation of bounded-deterministic functions (BDF) by circuits and formulas of the minimum size in arbitrary automaton bases is considered.

Without loss of generality we may consider the alphabet $\{0,1,\ldots,k-1\}$, $k \geqslant 2$, called the *k-alphabet*. A finite state automaton with the k-alphabet being its input and

☆ Translated from *Discrete* Analysis and Operations Research, Vol. 5(2), Novosibirsk, 1997, pp. 78–89.

[1] Supported by the Russian Foundation for Fundamental Research (Grant 97-01-01484).

E-mail address: orlov@rsuh.ru (V.A. Orlov).

0166-218X/$ - see front matter © 2002 Published by Elsevier B.V.
PII: S0166-218X(02)00306-2

output alphabet is called a *k-automaton*. By a *k-basis* we mean a finite system of k-automata to each of which a positive number (weight) is assigned. For simplicity, we shall consider initial automata with one output. In what follows, a basis is supposed to be a k-basis and a BDF is supposed to be a BDF over the k-alphabet.

A basis is called *complete* (functionally complete) if each BDF (each k-valued function, i.e., truth BDF) can be computed by a circuit in this basis.

In [4], an infinite sequence of complete bases was constructed for each $k \geqslant 2$. This sequence satisfies the condition that for each basis the asymptotics of the Shannon function is cH, where c is a constant depending on the basis, and H is a function depending on the number of implemented BDFs. There is no algorithm that allows, given a basis, to find the constant c, but the constant c can be found with an arbitrary accuracy.

Thus, in [4] it was proved that the problem of finding an asymptotic behavior of the Shannon function in the case of implementing a BDF by circuits in an arbitrary complete basis is algorithmically undecidable.

In [5] the implementation of a BDF by circuits in incomplete $(k+r)$-bases $(r \geqslant 1)$ was considered. In this case a recursive sequence M and its nonrecursive subsequence M_1 such that the asymptotic formula for the Shannon function is of the form H $(2H)$ were constructed in the case of bases from M_1 (from $M \setminus M_1$). The undecidability of this type is called the *strong algorithmic undecidability*.

In the present paper it is proved that for each $k \geqslant 2$ the problem of finding the asymptotic behavior of the Shannon function in the case of functionally complete bases (without extending the alphabet) is algorithmically undecidable in the strong sense.

We also compare the complexities of implementing k-valued functions by circuits and formulas in functionally complete bases.

The size of a circuit S is the sum of weights of its elements and is denoted by $L(S)$. By $L_B^{\mathrm{c}}(G)$ $(L_B^{\mathrm{f}}(G))$ denote the minimum size of a circuit (formula) in a basis B that implements a system of functions G. By $L_B^{\mathrm{c}}(k,n)$ $(L_B^{\mathrm{f}}(k,n))$ denote the maximum of $L_B^{\mathrm{c}}(f)$ $(L_B^{\mathrm{f}}(f))$, where f runs over all functions of the k-valued logic of n variables.

A basis B is called *c-regular* (*f-regular*) if

$$L_B^{\mathrm{c}}(k,n) \sim c_B^{\mathrm{c}} \frac{k^n}{n},$$

$$\left(L_B^{\mathrm{f}}(k,n) \sim c_B^{\mathrm{f}} \frac{k^n}{\log_k n} \right).$$

A basis is called *cf-regular* if it is c-regular and f-regular. A basis B is called *cf-equivalent* if it is cf-regular and $c_B^{\mathrm{c}} = c_B^{\mathrm{f}}$.

As proved by Lupanov [2,1], each 2-basis of functional elements is cf-equivalent. Below it is proved that for each $k \geqslant 2$, there exist cf-regular bases that are not cf-equivalent. We also prove that the recognition problem for the cf-equivalence of a basis in the case of functionally complete bases is algorithmically undecidable in the strong sense.

2. Simulation of deducing words in systems of homogeneous productions by means of finite automata

Here, we construct automata that simulate using *homogeneous productions* (see, for example, [3]).

A system of homogeneous productions T is given by an alphabet $A=\{a_1,\dots,a_h\}$, a positive natural number W (a step of the system), and a set of elementary transitions

$$a_i \to R_i \quad (1 \leqslant i \leqslant h), \tag{T}$$

where R_i is a word over the alphabet A.

To apply the T-production to an arbitrary word R over A means to delete the first w letters from the word R and to augment, on the right side, the word obtained by the word from the system (T) that corresponds to the first letter of the word R. The T-production is not applied to words shorter than W. Note that a system of homogeneous productions is a special case of systems of Post productions.

A word U is said to be T-deducible from a word V if there exists a finite chain $V, V_1,\dots,V_r, U$ of words such that each word is obtained by applying the T-production to the previous word.

The following claim is valid ([3]):

Claim 1. *There exists a system of homogeneous productions T and a word R_0 such that the set of all words that are T-deducible from R_0 is nonrecursive.*

Note that there exists a system of homogeneous productions with an algorithmically undecidable problem of deducibility when all words R_i are nonempty. In what follows we consider such a system of productions; a word that is deducible from a word R_0 is called *deducible*.

By $|Q|$ denote the length of a word Q; by Q^r, the concatenation of r words Q (the word Q^0 is supposed to be empty). An infinite sequence of letters is called *a superword*. By Q^∞ denote a periodic superword $QQ\dots Q\dots$.

We simulate deducing words in the system of homogeneous productions along the lines of [4,5]. However, the method of marking the first and the last letters in a word is also used because of more rigid requirements (for the basis to be functionally complete).

To this end, consider alphabets $A_b=\{a_1^b,a_2^b,\dots,a_n^b\}$ and $A_e=\{a_1^e,a_2^e,\dots,a_n^e\}$ corresponding to the alphabet A. Let R be an arbitrary word in the alphabet A ($|R| \geqslant 2$). Then the word R^{be} is assigned to R which is obtained from R by replacing its first and last letters a_i and a_j by the letters a_i^b and a_j^e, respectively. A superword of the form $\beta^v R^{be}\gamma^\infty$, where $v \geqslant 1$, is called a *v-code* of the word R over the alphabet A.

3. Main lemma

We now consider 2-automata. We shall code letters of the above alphabets by Boolean tuples of length $h+4$; to each letter we assign two tuples that differ in the last component.

We replace each letter of a v-code of a word R over A by any of its two codes. Each superword obtained this way is called a *v-binary code* of the word R.

Let D_0 be an autonomous automaton with one input that computes a binary code of a word R_0 for $v=1$ and such that the last components of codes for all letters are equal to 0. Let D be an automaton with one input that simulates applying the T-production described above, and E be an automaton with one state and two inputs that computes the Schäffer function $sh(x_1,x_2)=\bar{x}_1 \vee \bar{x}_2$.

The moments when the last components of the codes of letters are sent to the input of an automaton are called *s-moments*. Let E_R be an automaton with three inputs that computes the function $sh(x_2,x_3)$ for $t > h+4$ at the moments different from the s-moments. If a superword at the first input of the automaton is a binary code of a word R, then the automaton E_R computes the function $sh(x_2,x_3)$ at the s-moments. Otherwise, from the moment of distinction, the automaton E_R computes the constant 0 at all s-moments. Later on, we shall describe how the automaton E_R works for $1 \leqslant t \leqslant h+4$.

Let a 2-basis B_R that consists of the automata D_0, D, E of weight 2, and of E_R of weight 1 correspond to an arbitrary word R over the alphabet A. Note that all bases B_R are functionally complete and constitute a recursive set.

A BDF is called *autonomous* if its value does not depend on the values of the arguments.

Let R be an arbitrary word over the alphabet A. The following main lemma is valid.

Lemma 2. *An autonomous BDF can be implemented by a circuit in the basis B_R if and only if its output superword is a binary code of a deducible word.*

Before proving the main lemma, we describe the automata and the codes of letters; we also give the definitions and auxiliary statements.

4. Alphabets, codes of letters, and the automaton *D*

To describe the elements of the basis B_R, we introduce the following alphabets:

$$A_d=\{a_1^d,a_2^d,\ldots,a_h^d\},\ A^1=\{a_{1,1},a_{2,1},\ldots,a_{h,1}\},\quad A_b^1=\{a_{1,1}^b,a_{2,1}^b,\ldots,a_{h,1}^b\},$$
$$A_e^1=\{a_{1,1}^e,a_{2,1}^e,\ldots,a_{h,1}^e\},\quad A_d^1=\{a_{1,1}^d,a_{2,1}^d,\ldots,a_{h,1}^d\},$$
$$A_\beta=\{\beta,\beta_b,\beta_e,\beta_d\},\quad A_\gamma=\{\gamma,\gamma_b,\gamma_e,\gamma_d\},$$
$$A_0=A\cup A_b\cup A_e\cup A_d\cup A_\gamma,$$
$$A_1=A^1\cup A_b^1\cup A_e^1\cup A_d^1\cup A_\beta,$$
$$A^0=A_0\cup A_1.$$

We code letters of the above alphabets by Boolean $h+4$-tuples as follows. Two tuples that differ in the last component correspond to each letter. We now describe the other $h+3$ components of the codes of letters. A letter a_i $(1 \leqslant i \leqslant h)$ is coded by a Boolean

tuple $0^{i+2}10^{h-i}$. The code of the letter a_i^b (a_i^e, a_i^d) differs from the code of the letter a_i only in the second (the third, the second and the third) component. The letter γ ($\gamma_b, \gamma_e, \gamma_d$) is coded by a Boolean tuple 0^{h+3} (010^{h+1}, 0^210^h, 01^20^h). The codes of letters over the alphabet A_1 differ from the codes of the corresponding letters over the alphabet A_0 only in the first component.

The behavior of an automata will be given by means of a set of transitions of the form $q_ix \to q_jy$, where q_i, q_j are the states of the automaton, x is an input string, and y is an output string. The transitions have the following meaning. Let at some moment the automaton be at a state q_i and accept an input string x. Then according to the transition $q_ix \to q_jy$, the automaton produces the string y and transfers to the state q_j.

For brevity, we use the following conventions:

the absence of a symbol of an input string means that the right parts of all input strings are the same;

the record $q_M \Rightarrow q_jy$ denotes the set of transitions $q_ix \to q_jy$ for all $x \in M$;

the record $q_i \Rightarrow q_jy$ denotes the set of transitions $q_ix \to q_jy$, where x is the implicit input for the state q_i.

We also omit intermediate states and mainly describe the reaction of automata to the codes of letters from A^0. So, "a letter over the alphabet A^0" will mean "a code of the letter" in the description of automata and their properties.

The second (third) component in the codes of letters that is equal to 1 is called the *b-marker* (*e-marker*). By A_2 (A_3) denote the set of letters of the alphabet A^0 that have the *b*-marker (*e*-marker).

In the description of the automaton D we assume that $1 \leqslant i,\ j \leqslant h$ and $1 \leqslant r \leqslant w-2$. By R^e denote the word that is obtained by replacing the last letter a_j in the word R by the letter a_j^e.

The automaton D has the initial state q^0 and works in accordance with the following set of transitions:

$$
\begin{aligned}
&q^00 \to q_0^*0; \quad q^0\beta \to q_0\beta; \quad q^011 \to q_1^*11; \quad q^0 \Rightarrow q_0^*\beta;\\
&q_0^* \to q_0^*0; \quad q_1^* \to q_1^*1;\\
&q_0\beta \to q_0\beta; \quad q_0a_i^b \to q_{1,i}^1\beta; \quad q_0 \Rightarrow q_w^*\beta;\\
&q_{1,i}^rA \to q_{1,i}^{r+1}\beta; \quad q_{1,i}^r \Rightarrow q_{r+1}^*\beta;\\
&q_{1,i}^{w-1}A \to q_{2,i}\beta; \quad q_{1,i}^{w-1}A_e \to q_{1,i}^0\beta; \quad q_{1,i}^{w-1} \Rightarrow q_0^*\beta;\\
&q_{2,i}a_j \to q_{3,i}a_j^b; \quad q_{2,i}\gamma \to q_{4,i}\gamma_b; \quad q_{2,i}a_j^e \to q_{2,i}^0a_j^b;\\
&q_{2,i}\gamma_e \to q_{1,i}^*\gamma_b; \quad q_{2,i} \Rightarrow q_0^*\gamma;\\
&q_{3,i}a_j \to q_{3,i}a_j; \quad q_{3,i}a_j^e \to q_{2,i}^0a_j; \quad q_{3,i} \Rightarrow q_{4,i}\gamma;\\
&q_{4,i}A_3 \to q_{1,i}^*\gamma; \quad q_{4,i} \Rightarrow q_{4,i}\gamma.
\end{aligned}
$$

At state $q_{1,i}^0$ ($q_{2,i}^0$, $q_{1,i}^*$), the automaton D operates as an autonomous automaton with the output superword $R_i^{be}\gamma^\infty$ ($R_i^e\gamma^\infty$, $\gamma^{|R_i|-1}\gamma_e\gamma^\infty$).

At the state q_w^* (q_{r+1}^* $1 \leqslant r \leqslant w-2$) the automaton D operates as an autonomous automaton with the output superword $\beta^{w-1}\gamma^\infty$ ($\beta^{w-r-1}\gamma^\infty$).

It is easy to verify that the description of the automaton D is consistent. The behavior of the automaton D at Boolean strings of length $h+4$ that differ from the codes of letters of the alphabet A^0 is irrelevant. Therefore, the definition of the automaton can be extended consistently.

We next describe the functioning of the automaton D in the form of word transformations. We write the prefix of the input word on the left, and we write the output superword on the right.

By $a_{r,U}$ denote the rth letter of an arbitrary word U; by R (Q), an arbitrary word (possibly empty) over the alphabet A (over the alphabet $A^0 \setminus A_3$). Let R^- ($|R| \geqslant w$) be the word obtained from the word R by deleting the first w letters. By $R^{(r)}$ denote the word that is obtained from R by consecutively applying r T-productions (and putting $R^{(0)} = R$).

(1) $a(a \in A^0 \setminus \beta) \to 0^\infty,\ 1^\infty$ or $\beta\gamma^\infty$.
(2) $\beta^v a\ (a \in A^0 \setminus A_b) \to \beta^{v+w}\gamma^\infty$.
(3) $\beta^v a_i^b Ra\ (|R| < w-2, a \in A^0 \setminus A) \to \beta^{v+w}\gamma^\infty$.
(4) $\beta^v a_i^b Ra_j^e\ (|R| \geqslant w-2) \to (v+w)$–the binary code of the word $(a_i R a_j)^{(1)}$.
(5) $\beta^v a_i^b Ra\ (|R| = w-2, a \in A^0 \setminus (A \cup A_e)) \to \beta^{v+w}\gamma^\infty$.
(6) $\beta^v a_i^b R\gamma_e\ (|R| = w-1) \to \beta^{v+w}\gamma_b\gamma^{|R_i|-1}\gamma_e\gamma^\infty$.
(7) $\beta^v a_i^b R\gamma Qa\ (|R| = w-1, a \in A_3) \to \beta^{v+w}\gamma_b\gamma^{|Q|+|R_i|}\gamma_e\gamma^\infty$.
(8) $\beta^v a_i^b Ra\ (|R| = w-1, a \in A^0 \setminus (A \cup A_e \cup \gamma \cup \gamma_e)) \to \beta^{v+w}\gamma^\infty$.
(9) $\beta^v a_i^b Ra\ (|R| \geqslant w, a \in A_3 \setminus A_e) \to \beta^{v+w} a_{w,R}^b R^- \gamma^{|R_i|}\gamma_e\gamma^\infty$.
(10) $\beta^v a_i^b RaQe\ (|R| \geqslant w, a \in A^0 \setminus (A \cup A_3),\ e \in A_3) \to \beta^{v+w} a_{w,R}^b R^- \gamma^{|Q|+|R_i|+1}\gamma_e\gamma^\infty$.

The automaton E_R for $1 \leqslant t \leqslant h+4$ is given by

$$q_0(0,x_2,x_3) \to q_0^*0; \quad q_0^* \to q_0^*0; \quad q_0(1,x_2,x_3) \to q_1 sh(x_2,x_3).$$

At the state q_1 the automaton E_R computes the function $sh(x_2,x_3)$ for $2 \leqslant t \leqslant h+3$. If the string β is sent to the first input (the string other than β), then for $t = h+4$ the automaton E_R computes the function $sh(x_2,x_3)$ (the constant 0).

5. Properties of circuits in the basis B_R

A circuit in the basis B_R with n inputs and m outputs is defined as follows. We choose n poles (called the inputs of the circuit) and some collection of elements of the basis (possibly with repetitions). Each input of each element is connected either with the output of an element or with an input of the circuit. We choose any m elements. The outputs of these elements are called outputs of the circuit. Each input of the circuit is connected with an input of at least one element of the circuit. The output of each element (except for possibly an output of the circuit) is connected with an input of

some element. The outputs of elements that differ from the outputs of the circuit are called *internal nodes of the circuit.* And there is another restriction.

First, we consider circuits whose all elements are automata with one state (functional elements). We delete all insignificant inputs of all elements. The resulting circuit of functional elements cannot contain a cycle, i.e., a chain whose output is connected with an input of its first element.

The set of states of elements in a circuit is called the *state of a circuit.*

At each state a finite automaton computes a function. Thus, there exists a circuit of functional elements that corresponds to a circuit of "automatic" elements at each state. A circuit in an automaton basis is a circuit of "automatic" elements such that there is a circuit of functional elements that corresponds to this circuit of "automatic" elements at each state (that is accessible from the initial state).

It is known that each circuit in an automaton basis computes a BDF.

It is easy to check that at the initial state all elements of the basis B_R compute functions essentially depending on all their variables. Thus, no circuit in B_R contains cycles.

By an sD-chain ($s \geqslant 0$) we mean a circuit that consists of s automata D such that the output of each of them (except for the last) is connected with an input of the next automaton. A sD-chain for some s is called a D-chain. An input of a D-chain is an input of its first element, the rth output is the output of the rth element. We connect the output of the automaton D_0 with an input of the sD-chain. The circuit obtained is called an (D_0, sD)-chain.

Without loss of generality we can assume that each circuit in B_R contains only one (D_0, D)-chain.

From the functioning of automaton D as a word transformer, it follows

Lemma 3. *For $r \geqslant 0$ the superword at the $(r+1)$*th *output of the (D_0, D)-chain is a $(wr+1)$-binary code of the word $R_0^{(r)}$.*

The collection of components of a code from the 4th to the $(h+3)$th for a letter from A^0 is called its *stem*.

It is easy to check the validity the following Lemmas 3 and 4 concerning properties of superwords at outputs of the (D_0, D)-chains that simulate the deduction from a word R_0 in the system of homogeneous productions.

Lemma 4. *For any natural numbers p and s, the stems of the p*th *letters of the superwords either coincide or differ in only one component (depending on p) for all outputs of a (D_0, sD)-chain.*

Put $d_r = |R_0^{(r)}|$.

Lemma 5. *For any natural numbers s and $r \leqslant s+1$, the superword obtained at the r*th *output of a (D_0, sD)-chain has a pair of letters a_i^b and a_j^e at the $(wr+2)$*th *and $(wr+d_r+1)$*th *positions, respectively.*

Let B_E be a basis that consists of elements D_0, D, and E of weight 2. Consider properties of circuits in the basis B_E whose input superwords are 1-binary codes of the word R_0. These circuits are called (E, R_0)-*circuits*.

A maximal connected subcircuit that consists of elements E is called a *functional unit*. Note that the inputs of the functional unit are connected with the outputs either of automata D_0, or D, or else with inputs of the circuit; its outputs either are connected with the inputs of automata D or are outputs of the circuit.

We distribute the outputs of a functional unit and the D-chains among layers. The 0th layer contains the D-chain of a (D_0, D)-chain and the output of the functional unit such that all inputs of the functional unit on which it essentially depends are connected either with outputs of the (D_0, D)-chain or with inputs of the circuit. The ith level ($i \geqslant 1$) contains a D-chain whose input is connected with an output of a functional unit of the $(i-1)$th level and the output of a functional unit having at least one essential input connected with an output of an D-chain of the ith level, while its other inputs are connected with outputs of D-chains of the jth level ($j \leqslant i$) or with inputs of the circuit.

A Boolean function that preserves constants 0 and 1 is called an α-*function*. The output of the functional unit that computes an α-function is called an α-*output*.

By induction on the number of levels it is easy to check the following statement.

Lemma 6. *If an input of an element of a D-chain in the* (E, R_0)*-circuit is connected up to an output of a functional unit that is not an* α*-output, then the superword* 0^∞ *or the superword* 1^∞ *comes out at the output of this element.*

Therefore, in what follows we consider only those circuits in which the inputs of D-chains are connected to α-outputs.

The definition of an α-output implies

Lemma 7. *Let the inputs of a functional unit that are essential to an* α*-output of the unit receive words whose letters at the* pth position do not belong to the alphabet A_2 (A_3, A_1). *Then the letter at the* pth *position of the word computed at this output does not belong to the alphabet* A_2 (A_3, A_1), *too.*

Thus, a functional unit can neither create nor shift markers.

From Lemmas 4, 5, 7, the description of a functional unit D in the form of transformations of words, and the definition of an α-output, by induction on numbers of layers it is not difficult to obtain the following statement.

Lemma 8. *All letters of the word computed at an* α*-output of a* (E, R_0)*-circuit belong to the alphabet* A^0. *The output superword computed at the* α*-output either does not contain markers, or it includes markers by pairs, in which case the letters that contain these markers are at the* $(wr_i + 2)$th *and* $(wr_i + d_{r_i} + 1)$th *positions* $(r_1, r_2, \ldots$ *are appropriate numbers*). *For each* r, *the stems of letters with numbers from* $wr_i + 2$ *to* $wr_i + d_{r_i} + 1$ *either coincide with stems of letters in the word* $R_0^{(r_i)}$ *having the same numbers or coincide with the stem of the letter* γ.

6. Proof of main results

Proof of Lemma 2. The first statement of the lemma follows from Lemma 3. The possibility of computing binary codes for words over the alphabet A by a circuit in the basis B_R does not depend on the behavior of this circuit at s-moments. Since computing autonomous BDF is considered, to prove the second statement of Lemma it suffices to deal with (E, R_0)-circuits.

Let S be a (E, R_0)-circuit computing an autonomous BDF whose output superword is the binary code of a word P over the alphabet A not deducible from the word R_0.

It is easy to check that a superword which is computed at the output of the (E, R_0)-circuit and is not an α-output fails to be the binary code of a word over the alphabet A. Therefore, an output of the circuit S is either an α-output or an output of D-chain.

By Lemma 8, if a superword computed at an α-output is the binary code of a word Q over the alphabet A, then $Q = R_0^{(r)}$ for some r.

From the description of functioning the automaton D in the form of transformations of words, we see that if a superword at the α-output is not the binary code of a word over the alphabet A, then a superword that is computed at the output of a D-chain with an input connected with this α-output also fails to be the binary code of a word over the alphabet A. If a superword computed at the α-output is the binary code of a word Q over the alphabet A, then a superword that is computed at the pth output of a D-chain with an input connected with this α-output either is the binary code of the word $Q^{(p)} = R_0^{(r+p)}$, or is not the binary code of a word over the alphabet A. Lemma 1 is proved. □

Theorem 9. (a) *If a word R is deducible, then*

$$L^{c}_{B_R}(2, n) \sim \frac{2^n}{n}.$$

(b) *If a word R is not deducible, then*

$$L^{c}_{B_R}(2, n) \sim 2\,\frac{2^n}{n}.$$

Proof. In Case (a), we construct a circuit that computes an arbitrary Boolean function of n variables such that its output superword is the binary code of the word R. This construction is made from elements E_R whose first inputs are connected with an output of a (D_0, D)-chain.

In Case (b), let g be the number of elements D of a circuit S in a basis B_R that computes an arbitrary Boolean function of n variables. Let $W = \max_{1 \leqslant i \leqslant h} |R_i|$. It is easy to check that $d_g \leqslant (W - w)g$. Each automaton D can increase the number of initial letters β in the input word by w. Thus, from a number at most $wg + d_g \leqslant Wg$, all superwords computed at the outputs of the (D_0, gD)-chain consist of the letter γ only.

We assume that a prefix of the length Wg of each input word of the circuit S coincides with a prefix of a 1-binary word for a word R_0. Whether or not a word

is binary over the alphabet A can be determined before the first appearance of the letter γ.

By Lemma 2 it follows that Wg-prefixes of words at the first inputs of automata E_R differ from Wg-prefixes of the binary code of the word R. Therefore, for $t = (h + 4)j$, $j > Wg$, the automata E_R computes the constant 0.

Hence, the number of elements of E in the circuit S is asymptotically at least $2^n/n$. Theorem 9 is proved. □

Theorem 10. *For each word R over the alphabet A,*

$$L^{\mathrm{f}}_{B_R}(2,n) \sim 2\,\frac{2^n}{\log_2 n}.$$

Proof. Assume that a circuit S^{f} in the basis B_R computes a Boolean function of n variables, and an output of each element either is connected with an input of an element or is an output of the circuit.

The upper bound is obtained by constructing a circuit of functional elements E with the use of Lupanov's method [1].

We assume that all input words of the circuit S^{f} begin with symbol 0. Automata E_R whose first inputs are connected with an input of the circuit can be deleted because they compute constant 0.

Let the first input of the automaton E_R be connected with an output of an element K. If an input superword at the first input of the automaton E_R is not (is) a binary code of the word R, then we delete the element K and the automaton E_R (and replace the latter with element E). Thus, a circuit B_E is obtained from the circuit S^{f} such that the complexity of B_E is at most that of S^{f}.

The proof of the theorem is now completed by standard cardinality arguments. □

7. Conclusion

The analogs of Lemma 2 and Theorems 9 and 10 are valid for each $k > 2$. In order to prove upper bounds in the class of circuits (formulas), we apply a method from [5, p. 153]. Theorem 9 (and its analog) combined with Claim 1 imply the strong algorithmic undecidability of the problem of finding asymptotic behavior of the Shannon function in the case of functionally complete bases follows. Theorems 9 and 10 (and their analogs) along with Claim 1 yield the existence of cf-regular bases that are not cf-equivalent and the strong algorithmic undecidability of the problem of recognizing the cf-equivalence of the basis.

Acknowledgements

The author is grateful to O.B. Lupanov for discussing the results of the paper.

References

[1] O.B. Lupanov, On complexity of implementing Boolean functions by formulas, in: Problemy Kibernetiki, Vol. 3, Fizmatgiz, Moscow, 1960, pp. 61–80 (in Russian).
[2] O.B. Lupanov, On an approach to the synthesis of control system—the principle of local coding, in: Problemy Kibernetiki, Vol. 14, Nauka, Moscow, 1965, pp. 31–110 (in Russian) (see also Cybernetics collection. New series 11).
[3] A.I. Mal'cev, Algorithms and Recursive Functions, Nauka, Moscow, 1965 (in Russian) (see also Algorithms and Recursive Functions, Wolters-Noordhoff Publishing, Groningen, 1970).
[4] V.A. Orlov, Algorithmic unsolvability of the problem of asymptotic behavior of the Shannon function in realization of bounded-deterministic operators by circuits in arbitrary bases, Sov. Phys. Dokl. 16 (1971) 81–83 (translation from Dokl. Akad. Nauk SSSR 196 (1971) 1036–1039).
[5] V.A. Orlov, Complexity of implementing bounded-deterministic operators by circuits in automaton bases, in: Problemy Kibernetiki, Vol. 26, Nauka, Moscow, 1973, pp. 141–182 (in Russian).